

Année universitaire

2016-2017

**AUTEURS:**

Rémi BOUGUERMOUH

Anthony BERTRAND

Alexandra GONZALEZ

LP-MIAW

**MODULE MI6: architecture distribuée**

**RAPPORT DU PROJET**

# Objectif

L’objectif de ce projet est de créer un outil de chat en utilisant une architecture 3 tiers.

# Le dossier « Common »

Ce dossier contient les classes communes à tout le projet : Conversation, Message et User.

La classe Conservation est composée de la liste des messages constituant une conversation (messages).

La classe Message est composée de trois attributs : le contenu du message (content), l’utilisateur qui l’a rédigé (user) et la date de rédaction du message (date).

La classe User est composée de trois attributs : le login de l’utilisateur (login), le mot de passe de l’utilisateur (password) et un booléen permettant de savoir si l’utilisateur est connecté ou pas (isLogged).

Ces trois classes possèdent les méthodes getters et setters habituelles pour accéder aux attributs.

# Les services

On distingue trois types de services : ceux concernant l’authentification de l’utilisateur, ceux pour les données et ceux pour les utilisateurs.

Chaque service possède :

- une interface qui hérite de java.rmi.Remote. Cette interface ne contient aucune méthode mais indique simplement que l'interface peut être appelée à distance. Elle contient toutes les méthodes qui seront susceptibles d'être appelées à distance. De plus, la communication entre le client et le serveur lors de l'appel de la méthode distante peut échouer, c’est pourquoi chaque méthode doit pouvoir de lever l'exception java.rmi.RemoteException.

- une classe qui représente l'objet distant. Elle doit implémenter l'interface. Cette classe hérite de la classe UnicastRemoteObject qui contient les différents traitements élémentaires pour un objet distant dont l'appel par le stub du client est unique. Tout comme l’interface, l’appel des méthodes, mais aussi du constructeur peut échouer, c’est pourquoi chaque méthode, ainsi que le constructeur doit pouvoir de lever l'exception java.rmi.RemoteException.

**ChatAuth**

Interface ServiceAuth

Elle contient 4 méthodes :

- authentification () : permet à l’utilisateur de s’authentifier et de se connecter

- desauthentication () : permet à l’utilisateur de se déconnecter

- newUser() : permet à l’utilisateur de créer un compte

- isLogged() : Renvoie un booléen (true et false) et permet de savoir si l’utilisateur est connecté ou pas

Classe ServiceAuthImpl

Elle contient les quatre méthodes de l’interface qu’elle implémente, ici ServiceAuth, plus une autre méthode userExist() qui permet de contrôler si l’utilisateur (login et mot de passe) existe dans un fichier .txt qui joue le rôle de base de données.

**ChatData**

**Interface**

**ChatUser**

# Architecture du projet

Ce projet est divisé en trois tiers :

- tiers 1 : Représente le client

- tiers 2 : Récupère les messages en cours

- tiers 3 : Crée une instance du service authentification, puis la lie au registry qui se charge de l’envoyer au stub.

**Tiers 1 :**

Le tiers1 est composé d’une page auth.jsp gérée par une servlet HTTP (controler). Cette page permet à l’utilisateur de saisir son login et son mot de passe. A la validation du formulaire, le tiers 2 est consulté pour vérifier si le couple login/mot de passe est présent dans le fichier auth.txt représentant une pseudo base de données. S’il est présent, le serveur renvoie un statut 200.